METHOD AND SYSTEM FOR PERFORMING A MEMORY-MODE WRITE TO CACHE

FIELD OF THE INVENTION

[0001] The present invention relates to accessing cache memory and in particular to a method and system for performing a memory-mode write to cache memory.

BACKGROUND OF THE INVENTION

[0002] In many computer systems, a large amount of silicon area is devoted to cache SRAM for instructions or data. In existing designs, the cache SRAM performs standard cache functions, and additional SRAM is added if any other functions are desired. Such designs experience drawbacks incurred by this extra hardware. There is need for a cache memory that operates in multiple modes to provide enhanced functionality, thereby eliminating the need for additional memory and enhancing the cache functionality.

SUMMARY OF THE INVENTION

[0003] An embodiment of the invention is a method of writing to cache including initiating a write operation to a cache. In a first operational mode, detecting the presence or absence of a write miss and if a write miss is absent, writing data to the cache and if a write miss is present, retrieving the data from a further memory and writing the data to the cache. In a second operational mode, placing the cache in a memory mode and writing the data to the cache regardless of whether a write miss is present or absent.

[0004] Another embodiment of the invention is a system of writing to cache including a cache directory and a cache array. Control logic writes a valid field and an address to the cache directory and data to the cache array. The control logic includes hit miss complex logic for determining a compartment of the cache directory and the cache

array to be updated upon detecting a cache hit in a first operation mode. A least recently used (LRU) complex logic determines a compartment of the cache directory and the cache array to be updated upon detecting a cache miss in the first operational mode. The control logic determines a compartment of the cache directory and the cache array to be updated regardless of a cache hit or cache miss in a second operational mode.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIGURE 1 depicts an exemplary cache structure.

[0006] FIGURE 2 depicts exemplary control logic for performing a write operation to a cache directory.

[0007] FIGURE 3 depicts exemplary control logic for performing a write operation to a cache array.

DETAILED DESCRIPTION OF THE INVENTION

[0008] Embodiments of the invention include a cache controller that is able to re-use both the instruction and data L2 caches for other functions at certain times, thereby reducing hardware needed in a system. The L2 instruction and data caches may be used as static memory during normal system initialization to hold the firmware bootstrap module and work area.

[0009] The cache may be placed in memory mode based on a memory mode bit. Alternatively, the cache may dynamically change from memory mode to cache mode based on addressing. The controller for the cache is designed to respond differently to store operations when the address falls in a certain range. When the address is in the certain range, the cache is operating as a memory rather than as a cache, which is referenced as memory mode operation. Since a memory mode write updates the cache-directory with the contents of target address, subsequent load operations perform normally. When the address is outside the range, the cache operates in the manner of a

conventional cache. Using memory mode, the bootstrap module can be loaded into the instruction cache as if the cache were the target of a memory store operation. When the bootstrap module is executed, it can read/write data to the instruction cache as program storage, as well as using it as a source of instructions. When the initialization phase is complete, the instruction cache naturally reverts to normal operation referenced as cache mode. This eliminates the need for the development of a special purpose hardware state machine to initialize the cache, which would not be as flexible.

[0010] Figure 1 depicts an exemplary cache structure having a cache directory 10 and a cache array 12. In the embodiment shown in Figure 1, the cache size is 128 Kbytes and includes four compartments (associativity classes). The L1 line size (data unit transferred to and from the requesting processor) is 32 Bytes. The L2 line size is 64 bytes (data unit transferred to and from L3). The bus width for data transfer is 128 bytes. Hence, it takes two cycles to transfer an L1 line and four cycles to transfer an L2 line. Each directory compartment has two fields: the valid field and the address (or tag) field. The width of the cache compartment is 128 bits (also known as a quad word) and the depth (in the word direction) of the cache array is 2048 bits, four times deeper than the directory. It is understood that other cache structure may be used, and embodiments of the invention are not limited to the cache structure shown in Figure 1.

[0011] Operation of the cache in both cache mode and memory mode is described with reference to Figures 2 and 3. A first mode of operation is cache mode when a cache write is requested and a cache hit is detected. Figure 2 illustrates control logic for writing to the cache directory 10 in this mode. A processor issues a cache write command to the cache. Accompanying the processor write command (shown as signal write_cmd) is the internal address 16 which is latched by the L2 cache and shown as internal_addr(0:25) in Figure 2.

[0012] The high order bits of this latched internal address, hi_order_addr(0:14), are used by hit/miss complex logic 18 (HMCPX) together with the directory outputs to

determine whether there is a hit or miss. The directory entry is pointed to by bits 15:23, internal_addr(15:23). The hit/miss complex logic 18 includes directory arrays and a group of control logic. The timing diagram for a directory hit is shown in Figure 2. The processor issues a write command, shown as signal write_cmd. One cycle later a hit is detected by hit/miss complex logic 18. The L2 cache sends an address acknowledgment (not shown) back to the processor causing it to drop the write command signal. One cycle later a write directory pulse, shown as wrt_dir_1pulse, is high causing a cache hit write pulse, shown as cache_hit_wrt_1pulse to be active. The cache hit write pulse is asserted when there has been a hit. The notation "1pulse" is used to signify the pulse being one cycle wide.

[0013] Since there is a cache hit, one of the four compartment hit signals (shown as a_hit, b_hit, c_hit, and d_hit) will be raised and a pair of the compartment write pulses will be high to allow the valid field and the address field to be updated. For example, if compartment-A is hit, the signals write valid A (shown as wrt_val_a) and write address A (shown as wrt_addr_a) will both be high (depending on the status of a select all bins signal described in further detail herein) at the time when the cache hit write pulse is active. The write valid and write address signal are used to update the directory inside the hit/miss complex logic 18. The write valid signal updates the valid field with valid =1 and valid_p =0 at the output of gate 20 as shown in Figure 2. The address field is updated by the write address with bits 0:14 of the internal address (shown as hi_order_addr(0:14)) plus the associated parity bit from parity generator 22. It takes one cycle to update the directory fields.

[0014] Another mode of operation is cache mode when a cache write is requested and a cache miss is detected. In the case of cache miss, none of the compartment hit signals (a_hit, b_hit, c_hit or d_hit) will be on, but one of the four least recently used (LRU) signals (shown as a_lru, b_lru, c_lru, or d_lru) will be high based on an LRU algorithm implemented by LRU complex logic 24. The LRU signal is driven by a LRU

write pulse (shown as LRU_wrt_1pulse) which is active after the missed data has been brought back from L3 and stored into the L2 line buffer. The LRU write pulse and the LRU signals are generated by the LRU complex logic 24. The LRU complex logic 24 includes an LRU array, a change bit array and a group of complex control logic. Assuming the LRU complex logic 24 determines that compartment C is the least recently used one, a write valid c signal is generated which updates the valid field with valid =1 and valid_p = 0. The address field is updated by write address c with bits of the internal address (shown as hi_order_addr(0:14)) plus the associated parity bit from parity generator 22. It takes one cycle to update the directory fields. The exact arrival time of the LRU write pulse is determined by the response time of L3 and the switch element between L2 and L3.

[0015] Another mode of operation is memory mode when a cache write is requested. In memory mode, the updating of directory 10 is similar to that for cache hit mode. The write valid signal and write address are driven by the memory mode write pulse, shown as memory_mode_wrt_1pulse, which is not determined by a hit signal but by a memory mode bit shown as memory_mode. This memory mode bit can be set by device control registers, an array used to store configuration information. Alternately, the memory mode bit can be determined by address bits indicating that the cache will be in memory mode only for writing into certain predetermined address space (e.g., high order address bits equal to "1111"). The timing for a memory mode write is the same as that for the cache mode write when a hit is detected. The contents with which the directory fields are update are same as the previous two cases. In the cache hit mode and cache miss mode, the compartment to be updated is determined by hit/miss complex logic 18 and the LRU complex logic 24, respectively.

[0016] In the case of a memory mode write, the compartment to be written into is determined by the address itself. The bin identifier, shown as Bin_ID corresponding to bits 13:14 of the internal address 16 designates the cache compartment to be written into.

These two address bits are decoded into four control lines as shown in Figure 2. A select all bins bit, show as sel_all_bins, is used to select all bins in the cache directory 10. When this select all bins bit is set to 1, a memory mode write will result in selecting all compartments by ignoring bin identifier because the select all bins bit is connected to all four AND gates 26 as shown in Figure 2. The purpose of the select all bins bit is to invalidate all four compartments at the same time. When the select all bins bit is 1, the data to be written into directory valid field will be valid = 0 and valid_p = 1 as a result of inverter 28 and AND gate 20. Since the entries are to be invalidated, there is no point to update the address field and hence the write address signal is off by virtue of inverter 35. The select all bins function to invalidate all four compartments utilizes the memory mode logic and thus provides enhanced functionality with little or no additional logic.

[0017] Figure 3 depicts control logic for updating the cache array 12, which is similar to updating the cache directory 10. The difference is that the rather than a one cycle write directory pulse, a two cycle write cache pulse, shown as wrt_cache_2pulse, is used. A two cycle pulse is used because the data unit (L1 line size) includes two quad-words of 128 bits which is the basic transfer unit in this design example. Also, the LRU write pulse, shown as LRU_wrt_4pulse is a four cycle wide pulse because it takes four transfers to load an L2 line into the cache array in the event of miss. For a cache hit write or memory mode write, the data to be written in the cache array is supplied by the processor, the requester. For a cache miss write, the data to be written into the cache array is from the L2 line buffer which has been loaded with the data from L3. The data source selection is done by the miss detection signal, shown as miss_detect, applied to a data selector 34. When the select all bins bit is 1, the cache array needs no updating in memory mode because the directory entries are to be invalidated anyway. Cache array congruence is determined by bits 15:25 of internal address 16, internal_addr(15:25) and bin identifier determined by bits 13:14 of internal address 16, internal_addr(13:14) bits.

[0018] While the invention has been described with reference to an exemplary embodiment, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted for elements thereof without departing from the scope of the invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the invention without departing from the essential scope thereof. Therefore, it is intended that the invention not be limited to the particular embodiment disclosed as the best mode contemplated for carrying out this invention, but that the invention will include all embodiments falling within the scope of the appended claims.